université de BORDEAUX

# Projet Bases de données et Perl M1 Bioinformatique 2018/2019

PRÉVOT Yann, PIET Quentin

03 Avril 2019

# Sommaire

# 1 Introduction

## 1.1 Données mises à disposition et objectifs

Le but de ce projet est de mettre en place une base de données locale contenant des informations sur l'organisme suivant : *Arabidopsis Thaliana*.

### fichiers fournis

La base de données doit être construite à partir des données présentes dans deux fichier plats avec séparateurs. Le premier, issu de la base UniProt, comporte des informations sur les protéines référencées pour l'organisme d'intérêt. Un deuxième jeu de données issu de la base EnsemblPlants recense les genes identifés pour cet organisme ainsi que les réactions qui leur sont associées.

### Objectifs

Ce travail consiste à développer une application en Perl permettant de manipuler une base de données qui stocke les informations de ces fichiers, d'éxecuter des requêtes SQL sur la base de données créée et d'en afficher le résultat, soit dans le terminal, soit dans via fichier html (sous forme de tableau) consultable depuis un navigateur web.
Cette application se pésentera sous la forme d'un menu proposant différentes actions sur la base de données (ajout de données et questionnement de la base).

# 2 Création de la base de données et interactions

Que ce soit pour la création de la base ou son interrogation, Le module DBI, une interface générique permettant d'accéder à une base de données sera utilisée pour toutes les interactions avec celle-ci.

## 2.1 Structure de la base de données

Notre base de données est constituée de 4 relations (voir schéma ci-dessous). Ces relations sont en troisième forme normale et sont liées entre-elles par leurs clés primaires "Entry/UniprotId" qui correspondent à l'identifiant de la séquence protéique dans la base Uniprot.

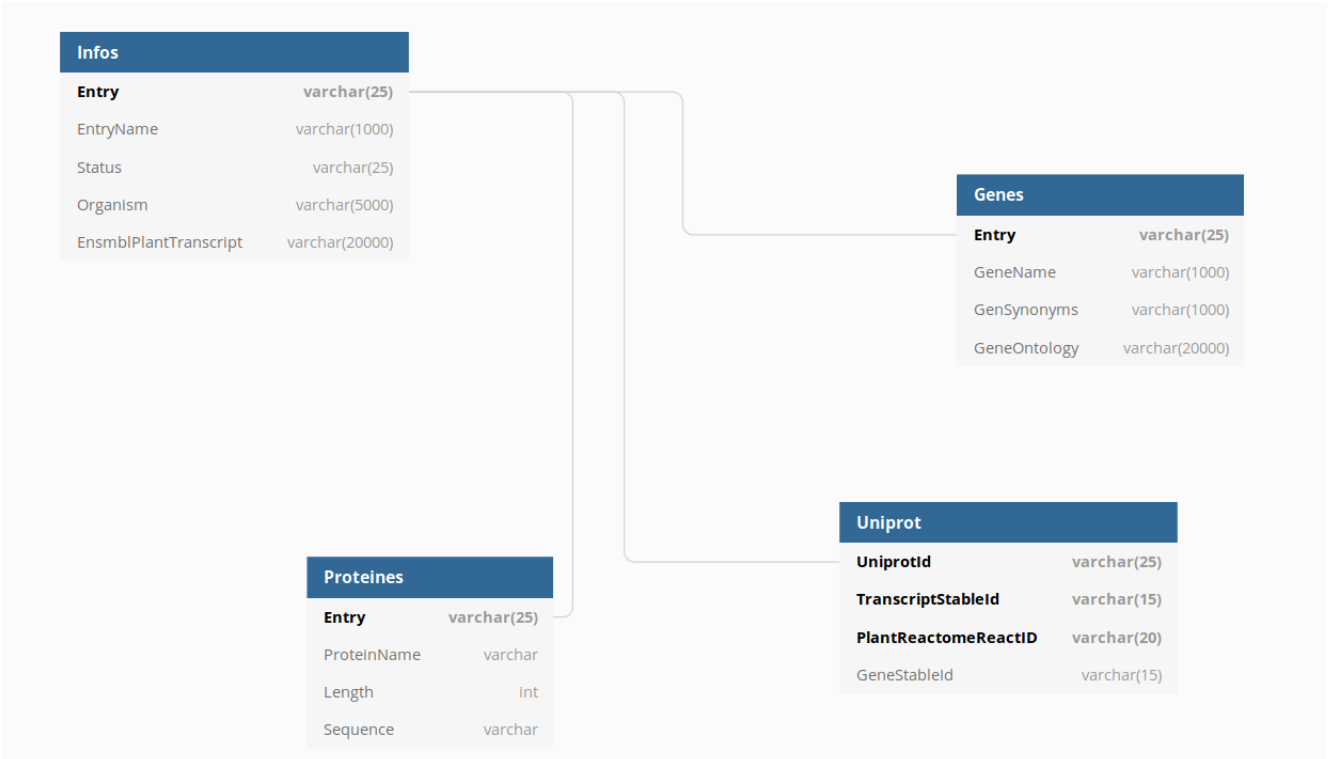


FIGURE 1 – schéma relationnel

## 2.2 Import des données dans les tables depuis les fichiers .csv

Les fichiers fournis pour la construction de la base de données sont des fichiers CSV (EnsmblPlant) et TAB (Uniprot). Ils possèdent tous les deux un séparateur de champs, respectivement la virgule et la tabulation.
On se connecte à la base de données à l'aide du module DBI. Le module Tie : :Handle : :CSV permet d'accéder facilement au contenu des fichiers texte à séparateurs. Il suffit de spécifier le caractère utilisé pour séparer les champs, et il est ensuite possible de spécifier le nom des

champs dans lesquels le script va itérer afin d'y récupérer les différents enregistrements. Ces valeurs seront ajoutées à la table correspondante grâce à une commande SQL (INSERT) préalablement préparée, et éxécutée à chaque itération.

## 2.3 Interactions entre l'utilisateur et la base de données

L'interface utilisateur se présente sous la forme d'un menu à choix s'affichant sur le terminal. L'utilisateur choisit quelle action réaliser en rentrant le caractère associé à cette action. Les requêtes sont également exécutées à l'aide du module DBI. Parmi les différentes options proposées à l'utilisateur, il peut soit ajouter des éléments dans la base (ajouter des protéines), soit en extraire des informations.

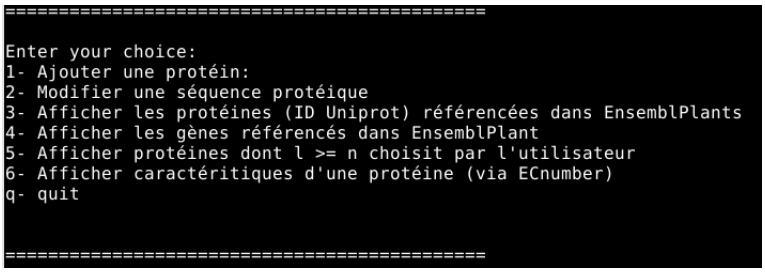


FIGURE 2 – menu de l'application

**Ajouter une protéines :**

Cette requête est effectuée par la fonction AddProtein() qui demande à l'utilisateur de rentrer le nom, l'identifiant Uniprot (entry) et la séquence de la protéine qu'il souhaite ajouter à la base. La longueur de la séquence est calculée et ces valeurs sont insérées dans la table PROTEINES grâce à l'expression SQL **"INSERT INTO <table>(<attributs>) VALUES (<valeurs>)"**.

**Modifier une séquence :**

L'utilisateur a la possibilité de modifier une séquence. Cette requête est réalisée par la fonction ModifSeq() qui demande à l'utilisateur de rentrer l'identifiant de la protéine (entry) dont il souhaite modifier la séquence. On cherche dans la table protéine si cette entrée existe. Si cette requête retourne un résultat, la fonction demande à lutilisateur de rentrer la nouvelle séquence. La nouvelle valeur de longueur est calculée puis les modifications dans la table PROTEINES sont effectuées via une expression SQL de type : **"UPDATE table SET colonne1 = valeur1, ... , colonnen = valeurn WHERE <ligne à modifier (entry)>"**.

**Afficher le nom des protéines qui sont référencées dans le fichier EnsemblPlant :**

Cette requête est effetuée par la fontion ProtEnsemblPlant();. Elle va chercher les enregistrements différents de l'attribut UniprotId dans la table UNIPROT (table créée à partir du

fichier EnsmblPlant). L'expression SQL utilisée est : **"SELECT DISTINCT UniprotId FROM uniprot"**

**Afficher le nom des gènes du fichier UniProt qui sont également référencés dans le fichier EnsemblPlant :**

Cette requête est effectuée par la fonction GenEnsemblPlant() qui va retourner les noms de gènes se trouvant dans le fichier uniprot (relation GENES ) et dans le fichier EnsmblPlant (relation UNIPROT) ainsi que l'identifiant <entry> de la protéine associée. Il s'agit des gènes dont l'identifiant de la protéine associée est le même.
La requête SQL utilisées : **"SELECT DISTINCT Genes.entry, Genes.GeneName FROM Genes JOIN Uniprot on Genes.entry = Uniprot.UniprotId"**.

**Afficher les protéines ayant une longueur au moins égale à une valeur donnée par l'utilisateur :**

La fonction SelectProtLength() demande à l'utilisateur d'entrer une valeur de longueur qu'il souhaite utiliser comme longueur minimale des protéines qui lui seront retournées.
La requête SQL utilisée : **"SELECT proteines.entry, proteines.length from proteines join infos on proteines.entry = infos.entry and organism = 'Arabidopsis thaliana (Mouse-ear cress)' WHERE proteines.length >= <ongueur entrée par l'utilisateur>"**.

**Afficher les caractéristiques de la ou les protéines correspondant à un *E.C. number* (dans le champs protein name) donné par l'utilisateur :**

Cette requête est réalisée par la fonction ProtCaract(). Elle demande à l'utilisateur de rentrer un *EC number* et va retourner les valeurs des attributs <entry> et <longueur> de la relation PROTEINES correspondant aux protéines trouvées avec le *EC number* dans le champs <proteinName> de la relation INFOS pour l'organisme 'Arabidopsis thaliana'.
La requête utilisée est la suivante : **" SELECT proteines.entry, proteines.length from proteines join infos on proteines.entry = infos.entry and organism = 'Arabidopsis thaliana (Mouse-ear cress)' where ProteinName ~ ':*$Ec_nb.*' "**.

# 3 Export des résultats de requêtes dans un fichier html

Les résultats des requêtes s'affichent d'abord à l'écran. Puis, dans le cas des trois dernières requêtes, les résultats sont également enregistrés dans un fichier html sous forme de tableau afin d'améliorer leur visualisation. Ce tableau est généré grâce à une fonction appellée après la réalisation de la requête et la récupération des résultats.

Dans un premier temps, les résultats et les attributs sont stockés dans une liste de type :
**@attributs = ( att1 , att2 );**
**@results = ( val1_att1 , val1_att2 , val2_att1 , val2_att2, ... , valn_att1, valn_att2 );**
*(exemple pour une requête retournant les résultats pour deux attributs).*

Ces listes sont ensuite passées en arguments dans la fonction permettant la construction du tableau html. À partir de la liste @attributs, le bon nombre de colonnes pourra être généré via une **"boucle for"**. Ensuite, une autre boucle permettra d'ajouter les résultats un à un dans les différentes colonnes et, à chaque pas **p = nb_attributs**, on génère une nouvelle ligne dans le tableau.

# 4 Conclusion

Le langage de programmation Perl, notamment grâce à l'utilisation de modules, se montre très utile pour la création, la gestion, et l'interrogation des bases de données. Les modules DBI et Tie : :Handle utilisé pour ce projet ont permit d'intégrer facilment du code SQL à nos script Perl et de traiter les données en amont afin d'importer les valeurs correspondant aux champs désirés depuis les fichiers CSV.

# 5 Annexes

## 5.1 Rendu Html

nb de lignes: 5

| entry | length |
|---|---|
| F4IA25 | 198 |
| Q9FNV6 | 181 |
| Q682L7 | 188 |
| A0A178WFM2 | 188 |
| Q8L8L7 | 188 |

nb de lignes: 95

| entry | length |
|---|---|
| Q8GY23 | 3681 |
| Q8H0T4 | 3658 |
| Q9SRU2 | 5098 |
| Q9M3G7 | 3856 |
| F4HZB2 | 3601 |
| F4JHT3 | 3527 |
| F4IG73 | 3001 |
| A0A1P8AUY4 | 5400 |
| F4IHS2 | 3574 |
| A0A384L2B6 | 3600 |
| A0A178VTN0 | 3543 |
| A0A1I9LQ67 | 3125 |
| F4JPL0 | 3804 |
| A0A1I9LSK5 | 5078 |

FIGURE 3 – rendu après enregistrment dans un fichier html